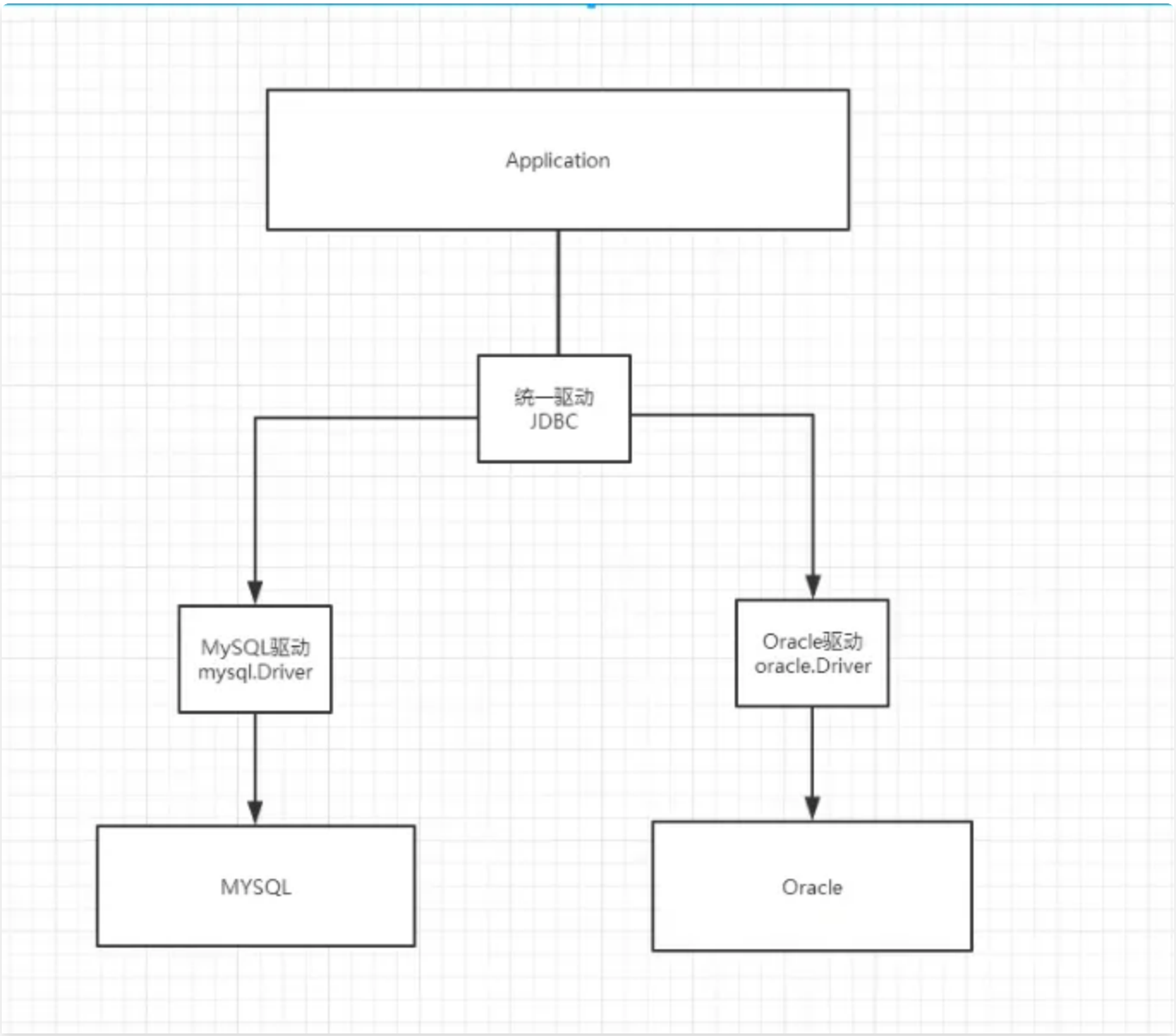
JDBC

什么是JDBC：java连接数据库



连接数据库需要jar包的支持：

- java.sql
- javax.sql
- mysql-conneter.....　　连接驱动，一定要导

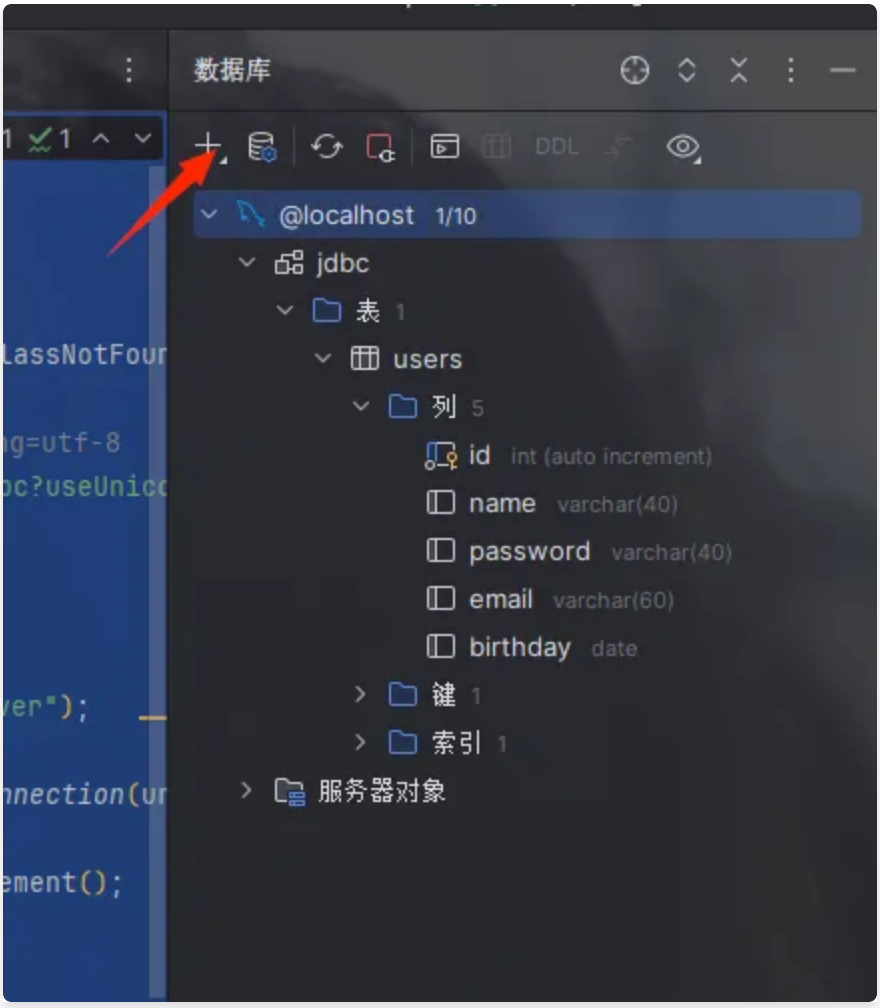
实验环境搭建

```sql
id=1
name=张三
password=123456
email=zs@qq.com
birthday=2000-01-01
id=2
name=李四
password=123456
email=ls@qq.com
birthday=2000-01-01
id=3
name=王五
password=123456
email=ww@qq.com
birthday=2000-01-01
```

导入数据库依赖

```xml
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.28</version>
</dependency>
```

IDEA中连接数据库



JDBC固定步骤:

1. 加载驱动
2. 连接数据库,代表数据库
3. 向数据库发送SQL的对象statement：CRUD
4. 编写SQL（根据业务，不同的SQL）
5. 执行查询SQL：返回一个ResultSet：结果集
6. 关闭连接，释放资源（一定要做）先开后关

```java
package com.kuang.test;

import java.sql.*;

public class Testjdbc {
    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        //配置信息
        //解决中文乱码useUnicode=true&characterEncoding=utf-8
        String url ="jdbc:mysql://localhost:3306/jdbc?useUnicode=true&characterEncoding=utf-8&useSSL=true";
        String username ="root";
        String password ="123456";

        //1.加载驱动
        Class.forName("com.mysql.jdbc.Driver");
        //2.连接数据库,代表数据库
        Connection connection = DriverManager.getConnection(url, username, password);
        //3.向数据库发送SQL的对象statement: CRUD
        Statement statement = connection.createStatement();
        //4.编写SQL
        String sql ="select * from jdbc.users";
        //5.执行查询SQL: 返回一个ResultSet : 结果集
        ResultSet rs= statement.executeQuery(sql);
        while((rs.next())){
            System.out.println("id="+rs.getObject("id"));
            System.out.println("name="+rs.getObject("name"));
            System.out.println("password="+rs.getObject("password"));
            System.out.println("email="+rs.getObject("email"));
            System.out.println("birthday="+rs.getObject("birthday"));
        }
        //关闭连接，释放资源（一定要做）先开后关
        rs.close();
        statement.close();
        connection.close();
    }
}

```

预编译SQL

```java
String sql = "insert into  users(id, name, password, email, birthday) values (?,?,?,?,?)";

//4.预编译
PreparedStatement preparedStatement = connection.prepareStatement(sql);

preparedStatement.setInt( parameterIndex: 1, x: 2);//给第一个占位符？ 的值赋值为1；
preparedStatement.setString( parameterIndex: 2, x: "狂神说Java");//给第二个占位符？ 的值
preparedStatement.setString( parameterIndex: 3, x: "123456");//给第三个占位符？ 的值赋值
preparedStatement.setString( parameterIndex: 4, x: "24736743@qq.com");//给第四个占位符？
preparedStatement.setDate( parameterIndex: 5,new Date(new java.util.Date().getTime()));

//5.执行SQL
int i = preparedStatement.executeUpdate();
```

事务

要么都成功，要么都失败

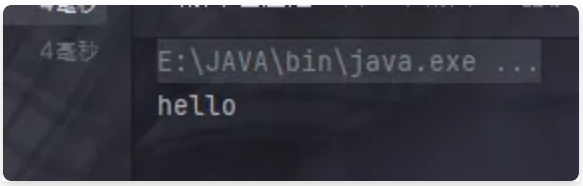
ACID原则：保证数据的安全

```Java
开启事务
事务提交   commit ()
事务回滚   rollback ()
关闭事务

转账:
A: 1000
B: 1000

A (900)  --100--》   B (1100)
```

junit单元测试

```XML
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
</dependency>
```

```java
public class Testjdbc3 {
    @Test
    public void test(){
        System.out.println("hello");
    }
}
@Test注解只有在方法上有效，只要加了这个注解的方法，就能直接运行
```



失败的时候：

搭建一个环境

```sql
CREATE TABLE account(
    id INT PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(40),
    money FLOAT
);

INSERT INTO account(`name`,money) VALUES('A',1000);
INSERT INTO account(`name`,money) VALUES('B',1000);
INSERT INTO account(`name`,money) VALUES('C',1000);
```

```sql
start transaction; #开启事务

update account set money = money-100 where name = 'A';

commit ;
```